METHOD FOR EXTRACTING CONTENT FROM STRUCTURED OR UNSTRUCTURED TEXT DOCUMENTS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority from U.S. Provisional Patent Application No. 60/263,574, filed on January 22, 2001, entitled "SYSTEM AND METHOD FOR DESIGNING, DEPLOYING AND MANAGING MOBILE APPLICATIONS."

FIELD OF THE INVENTION

The present invention relates generally to the endeavor of reusing or repurposing the contents of documents for use in other documents or applications. More particularly, the invention relates to a generic method for selecting/extracting a body of content from a textual document.

BACKGROUND OF THE INVENTION

The Internet has been a greatly successful medium that allows for the sharing of and access to essential information. This success also stems from the Internet's newfound ability to carry out transactions. Traditionally, the Internet has been accessed using web browsers running on personal computers linked to the Internet.

However, with the advent of new web technologies, users may now access the same information from a variety of different devices using disparate standards. The new devices not only run on different software systems than existing website and applications, they often use different mediums to transmit data, such as PSTN or wireless networks. More often than not, this makes such devices incompatible with existing sites. For example, a website built using HTML markup language and designed for personal computers using HTML-based browsers cannot operate with Internet-enabled, wireless phones that use Wireless Markup Language-based browsers.

In order to support these new devices and standards, a new breed of application will be built. A cost-effective solution for building these applications is to extract information from existing web sites, rather than implementing new systems from scratch. Thus, there is a need for a method to automatically extract information from current web sites and transform it for new application formats. This is referred to as *repurposing* content. Fundamental to this endeavor is the task of identifying the desired content or functionality within a web site for reuse.

A typical prior art approach for content and functional identification involves specifying the absolute location of the content, based on its location within the structure of the page's source code. However, this approach and others like it tend to be unreliable in practice, as web pages change in content and structure periodically. For example, a selection may be defined as, 'Select the third paragraph' for an HTML-based web page. As seen in Figure 1, this would result in the selection, "The quick brown fox slyly jumped over the lazy dog." However, if a new paragraph is inserted at the beginning of the document as seen in Figure 2, then the same selection definition, would yield "Starlight, starbright, first star I see tonight, I wish I may, I wish I might, have the wish I wish tonight."

Given that it is common for web pages to change in structure and content regularly, this problem suggests that a different and improved approach to identifying and selecting content from web pages and other computer-based documents is valuable. This method must be robust enough to operate successfully even after reasonable changes in structure and content. While the need for the present invention arose from work involving web sites and web applications, the invention is not limited exclusively to the domain of web sites and web applications. Numerous other applications will be apparent.

SUMMARY OF THE INVENTION

The invention presents a method to select content from text documents that may be extracted for use by other systems. A primary advantage of the present invention is that it selects content correctly and reliably from documents that may change in content or structure over time. Preferably, the present invention achieves content selection by applying a series of selection commands in succession. The selection commands successively narrow the scope of the selected content until the required content is reached. The selected content is said to be enclosed in a selection envelope. Selection envelopes are comprised of two virtual markers that delineate the boundaries of each envelope. An envelope is defined by positioning these virtual markers around a specified body of content in the document.

The definition of a selection envelope may be made relative to a previously defined envelope. This definition is based on various, non-limited means of identifying bodies of content or structures within a document. These means include, but are not limited to, computer-based functions and methods.

One non-limiting advantage of the invention is that it presents a method for defining selection commands for both structured and unstructured documents. Structured documents can be interpreted as having structural content and textual/character content. Unstructured documents can only be interpreted as having textural/character content.

Using a powerful and extensible command set, such as one described herein, it is possible for an operator to create robust selection commands that correctly function, even on constantly changing documents. The method is preferably embodied in a software-based development environment executing on a computer and manipulated by an operator. The operator may use this software to create a set of instructions for the selection of content from a given document. These instructions may then be executed by a computer-based, run-time entity to select a body of content. Once selected, the content may be 'repurposed' by other documents.

A non-limited series of selection commands may be defined for a document. Each successive command specifies a smaller envelope, or *child envelope,* defined relative to a preceding, or *parent,* envelope. Each successive command further "narrows" in on a desired body of content. In summary, this method of content identification is referred to as *Iterative Relative Enveloping* (IRE).

GLOSSARY OF TERMS

Begin marker: A virtual demarcation that signifies the commencement of a content envelope within the body of a web page.

Content selection envelope: See *Selection envelope.*

DTD: See *Structured document.*

End marker: A virtual demarcation that signifies the completion of a content envelope within the body of a web page.

Extraction command set: A set of selection envelopes. Applied to a set of source documents, an extraction set yields all the data to be extracted from the source for repurposing by another application.

IRE: See *Iterative Relative Enveloping.*

Iterative Relative Enveloping (IRE): An iterative process of selecting successively smaller envelopes of content. After selecting the first envelope of content, successive envelopes are all defined relative to the previous envelope.

Regular expression (regex): A pattern matching language to express how a computer program/human should look for a specified pattern in text. Regular expressions are composed of literal characters and metacharacters. Literal characters are normal text characters. Metacharacters combine literal characters according to a set of rules, similar to how arithmetic operators combine smaller (numeric) expressions.

Selection command: A function used to locate a specific piece of content within a document. If the content is located, begin and end markers may be placed adjacent to the content.

Selection envelope: A function of a set of domain-specific selection commands. The application of a selection envelope on a source document selects the desired data element(s).

Structured document: A structured document is a document whose contents follow a set of rules. Usually the rules are based on XML meta-language rules. XML is a World

Wide Web Consortium standard that allows other languages to be formally defined; it is not an application unto itself. Languages defined using XML meta-language rules are referred to as XML-conformant languages, or in short, XML languages. XML language rules are defined in two formats: Document Type Definition (DTD) or XML Schema Definition (XSD) format. A DTD is a set of rules governing the element types that are allowed in an XML document and the rules for specifying the allowed content and attributes of each element type. The DTD also declares all the external entities referenced within the document and notations that can be used. A schema definition is essentially equivalent to a DTD definition, with the additional ability to define the element and attribute types.

Unstructured document: Any text document. A stream of textual data does not need to follow any structural rules. One can treat a *structured document* as an unstructured document if needed.

Web application: See *Web site.*

Web page: A computer file that can be viewed by an end user in a web browser. These pages may be constructed in a variety of computer languages, such as HTML, WML, VoiceXML, XHTML, or any other suitable language. At present, HTML is the most prevalent source language for web pages.

Web site: A computer-based system of logical instructions, presentation files and data organized to form an interactive source of information accessible via computer networks.

XML: See *Structured document.*

The foregoing has outlined some of the pertinent aspects of the present invention. These aspects are merely illustrative of some of the more prominent features and applications of the present invention. Other benefits can be understood by applying the invention in a different manner or modifying the invention, as described below. These and other features and advantages of the present invention will be best understood from the following drawings and detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates the selection of the third paragraph of a HTML document.

Figure 2 illustrates the selection of the third paragraph of the HTML document in Figure 1, after a paragraph has been inserted.

Figure 3 is a flow diagram illustrating the process of repurposing content according to a preferred embodiment of the present invention.

Figure 4 illustrates the creation of a selection envelope by applying a selection command to a sample document.

Figure 5 illustrates the selection of an object in the structured hierarchy of a document.

Figure 6 illustrates the selection of content within a stream of content.

Figure 7 illustrates the relationship between multiple selection commands and selection envelopes, assuming every envelope is nested completely within its parent.

Figure 8 illustrates a child envelope that is relative to and nested within a parent envelope.

Figure 9 illustrates a child envelope that is relative to but only partially overlapping a parent envelope.

Figure 10 illustrates child envelopes that are relative to but outside parent envelopes.

Figure 11 illustrates the selection of two objects in the structured hierarchy of a document.

Figure 12 illustrates the selection of two strings within a stream of content.

Figure 13A illustrates the process of defining selection commands in a selection envelope to identify the desired content according to a preferred embodiment of the present invention.

Figure 13B is a flow diagram illustrating the creation of a selection command based on the document type and selection need.

Figure 14 illustrates the application of a selection command to select an object in the structured hierarchy of a document.

Figure 15 illustrates the application of a selection command to select a string within a stream of content.

Figure 16 is a viewable version of a sample web page, as rendered in a web browser.

Figure 17 is the HTML source for the sample web page in Figure 16.

Figure 18 illustrates a selection envelope surrounding the first table in the sample web page.

Figure 19 illustrates a selection envelope surrounding the second table in the sample web page.

Figure 20 illustrates a begin marker placed before the string "Section Title" and an end marker placed at the end of the document.

Figure 21 illustrates a selection envelope surrounding the first paragraph in the parent envelope shown in Figure 20.

Figure 22 illustrates how the sample page shown in Figure 21 may be altered without affecting the selected content.

Figure 23 illustrates the selection of the first table row containing the text "Row1."

Figure 24 shows an unstructured document in the form of a news story.

Figure 25 illustrates the begin marker placed behind the em dash and the end marker placed after the third paragraph in the example shown in Figure 24.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT OF THE INVENTION

The present invention provides a method for selecting content from within a document. In the preferred embodiment, the method may be implemented on a computer system, server, and/or software platform. Particularly, the method may be embodied within conventional software that may be implemented by at least one conventional computer system or network (e.g., a plurality of cooperatively linked computers). The system may be operatively and communicatively coupled to a computer network (e.g., the Internet), thereby allowing the method to operate over a network and select content from remote documents or files.

The discussion below describes the present invention in the following manner: (i) Section I provides a formulaic description on a general method for repurposing content according to a preferred embodiment of the present invention; (ii) Section II provides a definition for selection envelopes; (iii) Section III describes the concept of selection commands and how to create them; (iv) Section IV elaborates on the method using a structured document in HTML; and (v) Section V elaborates on the method using an unstructured example.

I. GENERAL METHOD OF REPURPOSING CONTENT

Figure 3 illustrates a method 1000 for repurposing content between two domains, according to a preferred embodiment of the present invention. A domain (Y) 1001 is an information source. If necessary, the information from domain (Y) 1001 is processed by a transformer (T1) 1002 into a set of textual documents. The information in domain (Y) 1001 may not be textual in origin, so that transformer (T1) 1002 may be required to convert it into text for use by the present invention. The present invention provides a method of selecting and extracting sets of information from text. This method is referred to as Iterative Relative Enveloping (IRE). These extracted sets of data are then passed to an external transformation system (T2) 1005. Transformation system (T2) 1005 may be required to convert the extracted data in a format that is used by a target domain (Y') 1006. Target domain (Y') 1006 may be any system that needs to use the information.

The elements in Figure 3 may be formally specified as follows: domain (Y) 1001 is the set of documents from the source domain; transformer (T1) 1002 is the external transformation system for transforming the source documents into text documents; selected data (X) 1004 is the set of data desired for extraction; transformer (T2) 1005 is the external transformation system for transforming the output data into the format needed for system (Y') 1006; and system (Y') 1006 is the target domain. In some cases transformers T1 and T2 could be null. Figure 3 illustrates the simplest case in repurposing Y for Y'. In practice, there can be any number of extraction sets and transformers between Y and Y'. The present invention provides a method to extract content from structured and unstructured documents using a set of extraction commands (E) 1003. This method is explained below using a series of equations as follows. When operated on domain (Y) 1001, E generates an extracted data set (X) 1004. This may be represented as follows:

(Equation 0) $E(Y) = X$

where E is the complete extraction command set, and is an unordered set of selection envelopes. It is called a selection envelope because it "selects" a portion of text each time it is applied to the source document set. Sets E and X may be defined as follows:

(Equation 1) $E = \{ s_1, s_2, s_3, \ldots, s_m\}$

where E is an ordered set of selection envelopes with cardinality m, and

each s_k is a selection envelope, $\forall k$ such that $0 < k \leq m$

(Equation 2) $X = \{ x_1, x_2, x_3, \ldots, x_m\}$

where X is an ordered set of all extracted data with cardinality m and x_k is the extracted data set, $\forall k$ such that $0 < k \leq m$

(The term ‘selection envelope’ is used in two ways. The first refers to a system of instructions that ‘selects’ content. The second refers to a container, or ‘envelope,’ generated by those instructions. This section uses the first definition. The second definition will be described below.)

The application of a selection envelope on the source document set results in a data element. More specifically, when any selection envelope s_k is applied to domain (Y), the result is the corresponding data element x_k.

(Equation 3) $s_k(Y) = x_k$

$\forall k$ such that $0 < k \leq m$

Selection envelopes are composed of instructions called *selection commands*. The set of all selection commands is typically domain specific. We denote the set of domain specific commands using the letter “C” as follows:

(Equation 4) $C = \{ c_1, c_2, c_3, \ldots, c_t\}$

where C is the set all of selection commands in a domain with cardinality t, and

c_k is a selection command, $\forall k$ such that $0 < k \leq t$

Each selection envelope is made up of a set of one or more selection commands or selection functions applied in sequence. A selection function is a meta-level command generated by combining various selection commands using logical or programming language constructs. Each selection envelope s_k may have a different number of selection commands, as required by the extraction. Each selection command in envelope s_k is an instantiation of a command in C, with parameters, or an instantiation of a function that is defined using the selection commands in C, with parameters. Thus, the number of selection commands n in an envelope s_k has no relation to t, the cardinality of C. A selection function "f" in equation (5) below is preferably a concatenation of one or more selection command instances.

(Equation 5) $s_k = f(C_k)$

where $C_k \subseteq C$

For any given selection envelope s_k using n selection commands, s_k contains n-1 envelopes, and the initial selection envelope is the same as the output of the initial selection command applied on the source document. Further, the selection commands are applied relative to the results of previous selection commands. The selection operator $\odot$ in equation (6) indicates the concatenation of any two selection two commands. For example, in a non-limiting embodiment $\odot$ could be one of "*" and "+," with "*" being similar to the Boolean "AND" operation, meaning "apply the previous command and this command," and " "+" being similar to the Boolean "OR" operation, meaning "apply the previous command, if false, then evaluate this command."

(Equation 6) $s_k^g = c_k^g (x_k^i) \odot s_k^{g-1}$

$\forall k$ such that $0 < k \leq m$,

where s_k^g is the envelope with g successively applied commands, $1 < g \leq n$,

c_k^g is the gth invocation of a selection command in set C_k,

x_k^i is the result of the ith selection command, such that $1 \leq i < g$, and

$\odot$ denotes the operation of selection. In the case where k=1,

(Equation 7) $s_k^1 = c_k^1 (Y) = x_k^1$

The first selection envelope is a result of applying the first selection command on the input set Y.

Note that, when expanded, equation (6) is also equivalent to (shown without parameters)

(Equation 8) $s_k^g = c_k^g \odot (c_k^{g-1} \odot(\ldots (c_k^2 \odot (c_k^1))\ldots))$

In equations (6), (7) and (8) each command c_k^g is the gth instance of command c_k or of function using commands defined in equation (4). These selection commands have required parameters that need to be specified when used. The same command may be applied multiple times in the same selection envelope with different parameters. For the sake of clarity, the parameters of the commands are not shown in the notation. Further, the notation c_k^g refers to the command of index g in any selection envelope s_k.

Applying each c_k^g successively on the previous selection envelope results in an intermediate data extraction, x_k^g. Note that c_k^g is an instance of a selection command, and may use any combination of the previously determined data sets, along with the initial input Y, as a parameter to the command. To elucidate further, the following shows all the intermediate steps in a selection of n steps:

$s_k^1 = c_k^1\ (f\{Y\})$ and $s_k^1(Y) = x_k^1$

$s_k^2 = c_k^2\ (f\{Y, x_k^1\}) \odot (s_k^1)$ and $s_k^2(x_k^1) = x_k^2$

...

$s_k^n = c_k^n(f\{ Y, x_k^1, \ldots, x_k^{n-1}\}) \odot (s_k^{n-1})$ and $s_k^n(x_k^{n-1}) = x_k^n$

Note that x_k^n, the result of the nth successive selection command, is the same as x_k, which is the required extraction data element.

By repeating the above to extract all the content specified by X, the set E is achieved.

This exemplary scenario is presented to further elucidate selection envelopes:

Let $\mathbf{Y} = \{ \mathbf{y_1, y_2, y_3, \ldots, y_{100}} \}$ be the source domain of HTML documents

Let $\mathbf{C} = \{\mathbf{c_1, c_2, c_3, c_4}\}$ be the set of available commands

where the command set is specifically defined as follows:

c1 selects the document y to operate in from the set Y

c2 is a regular expression pattern matcher

c3 selects tabular data

c4 returns list data

Suppose the goal is to extract

1. The table between “God .... value our own." in document **y12.**
2. The first list in the document **y25.**
3. The first table or first list of document **y4.**

Let $\mathbf{X} = \{\mathbf{x_1, x_2, x_3}\}$ represent the above three data sets,

and the operation $\odot$ is either “*”(logical AND) or “+”(logical OR).

The method specified herein may be used to determine the extraction command set **E** $= \{\mathbf{s_1, s_2, s_3}\}$ using the command set C.

The selection envelopes developed using this method are described below:

$\mathbf{s_1 = c_3' * (c_2' * c_1')}$; $C_1 = \{\mathbf{c_1, c_2, c_3}\}$

c_1’ selects document y_{12} from Y; it is an instantiation of c1

c_2’ parameterizes c_2 to only include content between “God .... value our own” in y_{12}

c_3’ further finds a table in between the scope “God .... value our own” in document $y_{12,}$ based on c_3

$\mathbf{s_2 = c_4' * c_1'}$; $C_2 = \{\mathbf{c_1, c_4}\}$

c_1’ selects document y_{25} from Y

c_4' further finds a list in document y_{25}

$s_3 = (c_{3'} * c_1') + (c_4' * c_1')$; $C_3 = \{c_1, c_2, c_3, c_4\}$

c_1' selects document y_4 from Y

c_4' further finds a list in document y_4

If the a list is available, it returns here, otherwise (the "+" operator)

c_1' selects document y_4 from **Y**

c_3' further finds a table in document y_4

These may be applied to input set Y such that

$s_1(Y) = x_1$

$s_2(Y) = x_2$

$s_3(Y) = x_3$

For any specific pair of domains, an extraction system consists of a design phase and an execution phase. During the design phase, an operator of the present invention uses the domain specific extraction commands C to produce an extraction command set E. This is achieved by defining specific selection envelopes to extract each data element. During the execution phase, a run-time system executes the selection envelopes to extract the contents.

II. SELECTION ENVELOPES

As mentioned before, selection envelopes are used both as instructions for selection of content and as a container for selected content. This second manifestation will now be described.

As shown in Figure 4, a selection envelope 1400 is a container for a section of a document, delineated by two markers referred to as the *begin marker* 1200 and *end marker* 1300. These markers are virtual delineators that are created only during runtime. The begin marker 1200 defines the beginning of the selection envelope 1400 while the end marker

1300 defines the end of the selection envelope. The selected contents 1500 is what lies between these two markers.

A selection envelope can contain elements from structured or unstructured documents.

For the purpose of this invention, it is assumed that all structured documents are based on XML meta-language rules. XML is a known World Wide Web Consortium standard. XML allows other languages to be formally defined; it is not an application unto itself. Languages defined using XML meta-language rules are referred to as XML-conformant languages, or in short, XML languages. XML language rules are defined in two formats: Document Type Definition (DTD) or XML Schema Definition (XSD) format. A DTD is a set of rules governing the element types that are allowed in an XML document and the rules for specifying the allowed content and attributes of each element type. The DTD also declares all the external entities referenced within the document and notations that can be used. Stated otherwise, an XML DTD provides a means by which an XML processor can validate the syntax and some of the semantics of an XML document. A schema definition is essentially equivalent to a DTD definition, with the additional ability to define the element and attribute types. XML based languages can be of two types, well formed and strict. Well-formed documents are structurally complete. Strict documents are always accompanied by a rule set (DTD or schema) and strictly follow those rules. This invention applies to both. An HTML document can be treated as a well-formed XML document and used in structural operations. In addition, structured documents have both structural and textual representations.

Unstructured documents, also known as 'character' documents, are textual documents and do not need to follow any structural rules. They are comprised of text symbols that can be of any type and can be ordered in any sequence. A structured document may also be treated as an unstructured document. ASCII text is an example of an unstructured document.

For structured documents, a selection envelope can contain various arrangements of structures. As shown in Figure 5, a structured document may be represented as a hierarchical structure 1110. A selection envelope 1410 made of a begin marker 1210 and end marker 1310 may contain any valid structural element represented object 1112. Selection envelopes containing structural objects place their begin markers and end markers immediate adjacent to the object so that they exclusively define the desired object. Just as the structure of a document may exist as an abstract system created by an XML processor, the begin and end markers are virtual objects in the document.

For unstructured documents, a selection envelope can contain contiguous segments of text based on the textual representation of the document. An example of a selection envelope with relation to an unstructured document is shown Figure 6. Begin marker 1220 and end marker 1320 are positioned around segments of content within the document.

More generally, a system of selection envelopes can be defined so that each successive selection envelope, or child envelope, is defined relative to a previously defined envelope, or parent envelope. As shown in Figure 7, selection envelope 1430 may be defined for source document 1100. Envelope 1430 may then be used to produce envelope 1431 via selection command 1602, and so on. Selection commands are more fully explained below.

The relationship between a parent envelope and its successor, or child envelope can take form in one of three ways. A child selection envelope 1441 may be either nested within a parent selection envelope 1440, as shown in Figure 8; partially overlapping a parent selection envelope, as shown in Figure 9; or completely outside of a parent selection envelope, as shown in Figure 10. The scope of the selection is iteratively refined until the desired content has been selected.

Furthermore, multiple sets of selection envelopes may exist simultaneously for a given document when a selection command is applied. Referring to Figure 11, a structured document 1110 can be seen to have two selection envelopes 1410 and 1411 that contain two different object structures. Referring to Figure 12, an unstructured document can be seen to also have two selection envelopes.

The means by which a selection envelope is defined may differ for each envelope in a set. Thus, while a parent envelope may be defined by associating a marker with a certain string, the child selection envelope may be defined by associating a marker with a structural object. The means by which selections are defined will be described in detail later.

The preceding discussion can be can be further illuminated by referring to Figure 13A. This figure illustrates the general process 2000 of creating a series of selection envelopes s_k^1 , s_k^2, ..., s_k^n for a document Y_k. It corresponds to equations (6), (7) and (8) described above.

The basic unit for this process is the specification of a selection envelope. Step 2004 specifies the source of information. A source may be a complete document or section of a document. For the first selection envelope, s_k^1, the source is the entire document Y_k. In step 2005, a selection command c is parameterized to operate on Y_k. In step 2006, parameterized command c_k^1 outputs data set x_k^1, which is the content selected by envelope s_k^1. Finally, step 2001 evaluates whether the desired content has been selected. If so, then x_k^1 is output to system Y' by way of Transformer T2. This completes the process. If the desired content has not yet been selected, then the specification of a second envelope s_k^2 begins. The source is the set containing document Y_k and the output of the previous selection command, x_k^1.

The process for the specifying s_k^1 is equivalent to equation (7) above.

Proceeding selection envelopes are specified using the same process described above. Like the first selection envelope, s_k^2 is defined by selection command c_k^2 that outputs x_k^2. At decision gate 2002, it is again evaluated if the desired content has been selected. If it has, x_k^2 is output to system Y' by way of Transformer T2. This is equivalent to equations (6) or (8) where g=2.

If the desired content has not yet been selected, further envelopes are defined until a final envelope s_k^n is defined. The source for s_k^n is the set containing source document Y_k and

$x_k^1, x_k^2, ..., x_k^{n-1}$. Selection command c_k^n outputs x_k^n, which is deemed to be the correct selection by the final decision gate 2003. This completes the process. This is equivalent to equations (6) or (8) where g=n.

With this understanding, the detailed workings of selection commands can now be explained.

III. SELECTION COMMANDS

As mentioned above, selection commands define selection envelopes or sets of selection envelopes. This section will describe the relationship between selection commands and selection envelopes.

For structured documents, the general relationship between selection commands and selection envelopes is illustrated in Figure 14. A selection command 1610 may identify an object structure composed of a child object 1112 and descendant objects 1113, and thus specify a selection envelope 1410 around the structure. For unstructured documents, this general relationship is illustrated in Figure 15. A selection command 1620 may define the locations of the virtual begin marker 1220 and virtual end marker 1320 and thus, define a selection envelope 1420.

Selection commands may use both structural and textual cues to define the selection envelope. Selection commands are not unique or universal; a set of extraction problems may require their own command set based on the markup language of the source document, a set of text operations, and programming language constructs.

Several prior art systems are based on either structure- or character-based operations. However, no prior art system has provided a combination of the two in the manner provided by the present invention, which offers increased flexibility. Also, the prior art systems based on structure-based operations use position-based information, such as second table, third paragraph, and others. The current invention creates selection commands based not only on position-based structure, but on semantic information (e.g., find the table with title "zzz") as well. Further, while most prior art methods enable automatic generation of selection commands, providing a method that uses human intervention during design time leads to more highly robust extractions. With human intervention, the selections can use intrinsic content markers in a document as part of the command that an automatic system could not.

Selection commands can be categorized into 3 different groups including (1) selection commands based on document structure; (2) selection commands based on character patterns or regular expressions; and (3) combined selection commands. Each of these groups is discussed below.

Group 1. Selection commands based on document structure

In any structured document, the "structure" is defined by notation that is interspersed among the document content. For example, in the case of XML-based documents, it is in the form of XML tags. These tags create a hierarchical structure. Thus, when these documents are manifested in memory, an operator can define several commands that capitalize on the document hierarchy. This typically results in a traversal of the non-linear data structures in memory. It is sometimes more optimal than using character-based operations.

For example, an XML document may contain a hierarchy of chapters, sections, and sub-sections. Once this has been read into memory, locating a certain paragraph of a certain section of a certain chapter becomes a trivial indexing location command. A character-based search, on the other hand, would perform a linear search.

The disadvantage of such indexing commands is the precise nature of the addressing. For documents that are periodically changing in structure, simply relying on structural commands may be disastrous, as illustrated in Figures 1 and 2.

The following table illustrates a few of the structure/context based selection commands on structured documents.

Structure/context based selection commands

Command name	**Example command instances**
Select elements by name	Given a name, select all the elements in the

	source document matching the name
Select element by location	Select elements by their location, such as the third table of the document, fifth address book entry, or nth occurrence of element k.
Select element by sibling relationship	Select the parent of element m with id = k or find the second sibling of element with id =k
Select element by attribute	Find all elements m whose attribute k has value v.
Select element by counter	Select nth child of root element.

Indexing commands such as “Select element by location” (e.g. find third table) may still successfully extract data when the structure of the document changes. However, contextual commands such as “Select elements by attribute” (e.g. find tables with title “Foobar”) will be more resilient to structural changes, assuming the content remains same even if the structure changes. If an XML document is not strict, then some of the contextual commands might not be useful, as the attributes specified by the commands might not be present in the source document. Then the most reliable way to identify them is to use structural commands. One of ordinary skill in the art will appreciate how to create or extend more context/based selection commands based on element order, attributes, and various relationships.

Group 2. Selection commands based on character patterns or regular expressions

Pattern- or regular expression-based operations treat text documents, both structured and unstructured, as a stream of characters, ignoring any structural notation that may be interspersed in the document. Most of these commands use patterns in the content itself to identify regions of text. By applying formal language theory, an operator of the present invention may build powerful regular expression commands to search and operate on bodies of text.

To create pattern-based selection commands, the input (or the contents) of the envelope is considered to be a stream of characters with certain delimiters such as 'space', 'comma', 'newline', and others. Those skilled in the art can appreciate how to create commands using regular expressions to find text containing specified strings and regular expressions. The table below illustrates two such operations:

Character-based selection commands

Command name	**Example command instance**
Select text contain	Select text containing the word 'patents'
Select text matching pattern	Select text matching pattern [1-9][0-9]*(\[0-9][0-9])?

The present invention allows selection commands to define the position of one or more pairs of virtual begin and end markers within a document, and in so doing, defines a selection envelope or a set of selection envelopes.

Group 3. Combining context- and pattern-based selection commands using programming language constructs

Selection commands that have literal interpretations, such as "Select the third table after the statement 'Final report:'" or "Select the table with the string 'Stock Symbol:' anywhere in the first row," are compositions created using both structural- and character-based concepts. These are the most flexible and robust commands.

Those skilled in the art will appreciate how to use programming constructs such as conditionals, loops and variables, in addition to the two types of selection commands described above, to create meta-selection commands. For example:

```
Var  k = (Select element 'i' with id='z')  ;
Result  := for each element e in k  do;
```

```
            if e contains the pattern 'text'
                    Select e;
    End-if
      End-for each
```

will select all elements with id 'z' and containing pattern 'text' in the source document.

Designing selection commands for documents

The general process for creating selection commands is shown in Figure 13B. This process provides for the creation of selection commands and selection functions.

In step 2016, the source document is evaluated automatically by a system or manually by an operator to be either a structured or unstructured document. Based on this, step 2017 is to select and parameterize an appropriate selection command. For structured documents, these include but are not limited to structural/contextual selection commands 2013 and pattern-based selection commands 2015. For character-based documents, this includes but is not limited to pattern-based selection commands 2015. Furthermore, the command set for structured documents may be based on combinations of structure/context-based selection commands and pattern-based selection commands. This is accomplished with the use of programmatic language constructs 2014. Programming language constructs 2014 may also be used to enhance selection commands for all documents by providing the ability to add conditions, loops, branching and other constructs.

The output of this process is a parameterized selection function c_k^n, which will create a selection envelope s_k^n, similarly to equation (6) above.

IV. EXAMPLES OF THE OPERATION OF METHOD 2000 USING A STRUCTURED DOCUMENT

An application of the present invention is illustrated in the following examples, using a structured document, or more specifically, a web page based on HTML. The examples illustrate the general process as shown in Figure 13A of extracting data from source Y for use in system Y'. They also illustrate the ability to create robust selection commands via the process shown in Figure 13B.

The method 1000 will be defined as follows for the following four examples. The source document Y is an HTML document, seen in rendered form in Figure 16 and in HTML source view in Figure 17. The examples will illustrate the creation of four selection envelopes s_1, s_2, s_3, and s_4 that respectively identify x_1, x_2, x_3, and x_4. As described above, selection envelopes are functions of selection commands 'c' that are defined below.

Specifically, the content selection goals for this example are as follows: Selection envelope, s_1, is to contain x_1, the first table in source document Y. Envelope s_2 is to contain x_2, the second table in the document. Envelope s_3 is to contain x_3, a certain paragraph in the document specified in detail below. Lastly, s_4 is to contain x_4, a certain paragraph containing a given string specified in detail below.

For the purposes of these examples, an initial selection envelope exists before any selection commands are specified. This envelope contains the entire source document.

Let

$\mathbf{s_1} = f(\mathbf{c_1})$

$\mathbf{s_2} = f(\mathbf{c_1})$

$\mathbf{s_3} = f(\mathbf{c_2}, \mathbf{c_1})$

$\mathbf{s_4} = f(\mathbf{c_3})$

Let X = { $\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$, $\mathbf{x_4}$ } represent the result of the above three selections where

$\mathbf{s_1}$ yields $\mathbf{x_1}$

$\mathbf{s}_2$ yields $\mathbf{x}_2$

$\mathbf{s}_3$ yields $\mathbf{x}_3$

$\mathbf{s}_4$ yields $\mathbf{x}_4$

when applied to source document Y.

Let $\mathbf{E} = \{ \mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4 \}$ where E is the complete set of content selected by the selection commands.

For the purposes of this example, let the total set of selection commands used be $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3\}$, where

$\mathbf{c}_1$ is a structural selection command with parameters:

type - the type of structural object to select; values can be HTML tag set

instance - the index of occurrence of the type of structure

inclusion - governs if the identified content is included or excluded

$\mathbf{c}_2$ is a pattern matching selection command that positions the begin and end marker with parameters:

begin marker string - the text string to be located

begin marker instance - the index of occurrence of the text string

begin marker inclusion - governs if the identified content is included or excluded

end marker string - the text string to be located

end marker instance - the index of occurrence of the text string

end marker inclusion - governs if the identified content is included or excluded

$\mathbf{c}_3$ is both a structural and pattern matching selection command that finds a structure based that contains a certain string. Its parameters are:

type - the type of structural object to select; values can be HTML tag set
instance - the index of occurrence of the type of structure
string - the text string contained in the structural object
inclusion - governs if the identified content is included or excluded

Now that the system has been defined, selection envelopes s_1 , s_2, s_3, and s_4 are now defined per the process illustrated in Figure 13A and Figure 13B. Relevant benefits of the invention will also be pointed out.

A. Selection Envelope s_1

This selection envelope example illustrates the ability to directly identify a structural object within a document by its position or sequential index with reference to a parent selection envelope. Referring to the process seen in Figure 13A, step 2004 is to define the source information for envelope specification; the source is document Y. For step 2005, a selection command c_k^1 is to be selected from the set of functions C defined above and then parameterized.

This calls steps 2016 and 2017 of the process in Figure 13B. Given that document Y is structured, step 2016 allows structural, pattern-based, or any combination of selection commands c_1, c_2 , or c_3 to be used. For the purposes of the example, the desired content x_1, which is the first table in the source file Y, is deemed to be reliably extractable by immediately using a single structural selection command c_1. Thus for step 2017, a structural selection command $\mathbf{c_1}$ is chosen and parameterized as follows:

type = table

instance = 1

inclusion = true

Thus, the output in step 2018 is $\mathbf{c}_1$ such that

$\mathbf{c}_1$ defines a resulting selection envelope, $\mathbf{s}_1$. This is represented as:

$s_1 = f(\mathbf{c}_1)$

which is equivalent to equation (5) above. Stated another way,

$s_1 = \mathbf{c}_1(\mathbf{Y}) = \mathbf{x}_1$

where $\mathbf{x}_1$ = the first instance of a table in document Y.

As shown in Figure 18, this command places begin marker 3012 and end marker 3013 so that they immediately surround the HTML table structure of the first table. As the desired content has been selected, the answer for step 2001 is 'yes' and the selected content $\mathbf{x}_1$ is available for use in Y'.

B. <u>Selection Envelope s_2</u>

To further elaborate on the use of selection commands, the second table of document Y will be selected for use in Y'. This again illustrates the use of position or sequential index or an object within a parent selection envelope.

Again utilizing the process seen in Figure 13A, step 2004 is to define the source information Y. The desired content x_2, the second table in the source file Y, is deemed to be reliably extractable by immediately using a single structural selection command c_1. Thus, for step 2005, selection command c_1 is selected for parameterization:

type = table
instance = 2
inclusion = true

Thus, the output of $\mathbf{c_1}$ is such that

$$s_1 = f(\mathbf{c_1})$$

which is equivalent to equation (5) above. Stated another way,

$$s_2 = \mathbf{c_1}\ \mathbf{(Y)} = \mathbf{x_2}$$

where $\mathbf{x_2}$ = the second instance of a table in document Y.

This selects the second table, as shown in Figure 19. As the desired content has been selected, the answer for step 2001 is 'yes' and the selected content x_2 is available for use in Y'.

C. Selection Envelope s_3

This next selection envelope example illustrates the ability to use multiple selection commands in series to define a selection envelope for a source document that may change in structure or content. This example also illustrates that different types of selection commands can be specified within the same selection envelope as necessary. Utilizing the process seen in Figure 13A, step 2004 is to define the source information for envelope specification; in this case, Y.

To develop the desired selection command, the process of Figure 13B is followed. Step 2016 dictates that either structural, pattern-based or any combination of selection commands c_1, c_2, or c_3 can be used. For the purposes of the example, the desired content $x_3{}^1$ which is the first paragraph after the string, "Section Title," in the source file Y, needs two selection commands for reliability, given that source document Y may change. For step 2017, the first selection command is determined to be a pattern-based selection command 2015, as seen in Figure 13B. Command $\mathbf{c_2}$ is chosen and parameterized as follows:

begin marker string = "Section Title"
begin marker instance = 1
begin marker inclusion = true
end marker string = end of document
end marker instance = n/a
end marker inclusion = n/a

Thus, the output $\mathbf{c_2}$ is such that:

$$s_3{}^1 = f(\mathbf{c_2})$$

which is equivalent to equation (5) above. Stated another way,

$$s_3{}^1 = c_2(Y) = x_3{}^1$$

where $\mathbf{x}_3^1$ can be seen in Figure 20.

Referring to step 2001, the desired content has not yet been selected thus necessitating the definition of another selection envelope. For this second selection envelope, the source in step 2004 document Y and $\mathbf{x}_3^1$. For step 2005, selection command c_k^1 has not yet been chosen. To determine c, the process of Figure 13B is again followed. Step 2016 dictates that either structural, pattern-based or any combination of commands c_1, c_2, or c_3 can be used.

For step 2017, the first selection command is determined to be a structural selection command 2013, as seen in Figure 13B. Command $\mathbf{c}_1$ parameterized as follows:

type = table
instance = 1
inclusion = true

Thus, $\mathbf{c}_2$ is such that:

$$s_3^2 = f(\mathbf{c}_1)$$

which is equivalent to equation (5) above. Stated another way,

$s_3^2 = c_1(x_3^1) \odot s_3^1 = x_3^2$ where $\mathbf{x}_3^2$ can be seen in Figure 21 where the begin marker 3003 and end marker 3004 surrounding the first HTML paragraph in the parent envelope. This is the desired selection $\mathbf{x}_3^2$. Furthermore, according to step 2002 in Figure 13A, no further selection envelopes need to be defined.

The robustness of selection envelope s_3 is illustrated by showing that it still correctly extracts the desired content from an altered source document. The original HTML source document is shown in Figures 16 and 17. The altered HTML source document 3007 is shown in Figure 22. Specifically, a paragraph 3008, horizontal rule 3009 and table 3010 have been added. The string "Section Title" now resides within table 3010. While these alterations have been made to the source page, the selection command defined for s_3^1 still successfully positions the begin marker 3011 and end marker 3012, for the first selection envelope. Similarly, the selection command defined for s_3^2 successfully positions the begin marker 3013 and end marker 3014, for the second selection envelope.

D. Selection Envelope s_4

This selection envelope example illustrates the use of a command that combines structural and pattern-based command. Yet again, the process of Figure 13A is used. Step 2004 defines the source information for envelope specification; in this case, the source is document Y, as shown in Figure 17. For step 2005, a selection command c_k^1 is to be selected from the set of functions C defined above and then parameterized.

In order to do this, steps 2016 and 2017 of the process in Figure 13B are used. Given that document Y is structured, step 2016 of the process seen in Figure 13B allows either structural, pattern-based or any combination of commands c_1, c_2, or c_3 to be used. For the purposes of the example, the desired content x_4, is deemed to be reliably extractable by immediately using a selection command c_3. Command c_3 combines structural and pattern-based commands using programmatic constructs. Thus for step 2017, both a structural/contextual selection command 2013 and a pattern-based selection command 2015 are selected. The selection command$\mathbf{c_3}$ is parameterized as follows:

type = row
instance = 1
string = "Row1"
inclusion = true

Thus, $\mathbf{c_3}$ is such that

c_3 defines a resulting selection envelope, $s_4$such that:
$s_4 = f(c_3)$
which is equivalent to equation (5) above. Stated another way,
$s_4 = c_3\ (Y) = x_4$

where $\mathbf{x_4}$ can be seen in Figure 23. As the desired content has been selected, the answer for step 2001 is 'yes' and the selected content x_4 is available for use in Y'.

As shown in Figure 23, the begin marker 3005 is placed before the opening structural tag for a table row <tr>, and end marker 3006 is placed immediately after the closing tag </tr> for the same table row. This is the selected and outputted content $\mathbf{x_4}$ equivalent to item 1551 in Figure 13A As specified by the command, table row contains a cell with the text "Row1" inside it.

V. AN EXAMPLE OF THE OPERATION OF METHOD 2000 USING AN UNSTRUCTURED DOCUMENT

The present invention can also be applied to non-structured documents. The following is an example of the use of the invention to extract content from a non-structured document as can be seen in Figure 24. The source document Y 4000, is a news story. The desired content from the document 4000 consists of only selection: the first three paragraphs.

The following example will be explained in reference to the extraction process illustrated in Figure 3 and the equations in Section I above.

The source domain Y for the system consists of document 4000. An extraction set E can be immediately applied to document 4000, as a transformer T1 is not required to transform the source into text. E is defined in order to produce the desired data set X. In this case,

$X=\{ x_1 \}$

where x_1 is the complete set of extracted data from document 4000.

The data set x_1 possesses one member element, a string containing the first three paragraphs of the news story in document 4000.

In order to extract set x_1, a selection envelope s_1 must be applied to document 4000.

From equation (5), it follows that $s_1 = f(C_1)$ where C_1 is a subset of all the selection commands in the current domain C.

Let $\mathbf{C} = \{ \mathbf{c_1} \}$, the total set of selection commands used, where

$\mathbf{c_1}$ is a pattern matching selection command that positions the begin and end marker with parameters:

begin marker string - the text string to be located

begin marker instance - the index of occurrence of the text string, and

begin marker inclusion - governs if the identified content is included or excluded
end marker string - the text string to be located
end marker instance - the index of occurrence of the text string, and
end marker inclusion - governs if the identified content is included or excluded

Utilizing the process seen in Figure 13A, step 2004 is to define the source information for envelope specification; in this case Y_k 1151 is document Y 4000. For step 2005, a selection command c_k^1 1651 is to be selected from the set of functions C defined above and then parameterized.

In order to do this, steps 1 through 3 of the process in Figure 13B are run through. Given that document Y 4000 is structured, step 2011 of the process shown in Figure 13B allows either structural, pattern matching or any combination of selection commands c_1, c_2, or c_3 can be used. For the purposes of the example, the desired content x_1 which is the first table in the source file Y, is deemed to be reliably extractable by immediately using a single structural selection command c_1. Thus for step 2015, the structural selection command 2013 is selected. This selection command is chosen to be $\mathbf{c_1}$ and parameterized as follows:

begin marker string = "—"
begin marker instance = 1
begin marker inclusion = false
end marker string = ".¶"
end marker instance = 3
end marker inclusion = true

This allows for step 2018 which defines c_k^n 1652 equal to c_1 such that

$\mathbf{c_1}$ defines a resulting selection envelope, s_1 such that:

$s_1 = f(\mathbf{c}) \text{ where } c = \{\mathbf{c_1}\}$

which is equivalent to equation (5) above. Stated another way,

$s_1 = \mathbf{c_1}\,(\mathbf{Y}) = \mathbf{x_1}$

where $\mathbf{x_1}$ can be seen in Figure 25.

As seen in Figure 25, $\mathbf{c}_1$ places the begin marker 4002 after the em dash 4001. $\mathbf{c}_1$ also places the end marker 4003 after the carriage return following the third paragraph. The selected content is x_1.

Thus, after applying one selection command described above to system Y, the selection function $f(C_1)$ yields the desired data set x_1. This data may now be passed to transformer T2 to be converted to a format appropriate for any target domain Y'.

It should be understood that the inventions described herein are provided by way of example only and that numerous changes, alterations, modifications, and substitutions may be made without departing from the spirit and scope of the inventions as delineated within the following claims.